A Generic Architecture for Redesign of Organizations triggered by Changing Environmental Circumstances

Hoogendoorn, M.; Jonker, C.M.; Treur, J.

published in
Computational and Mathematical Organization Theory
2011

DOI (link to publisher)
10.1007/s10588-011-9084-8

document version
Publisher's PDF, also known as Version of record

Comput Math Organ Theory (2011) 17: 119–151
DOI 10.1007/s10588-011-9084-8

A generic architecture for redesign of organizations triggered by changing environmental circumstances

Mark Hoogendoorn · Catholijn M. Jonker · Jan Treur

Published online: 12 February 2011

Abstract Artificial Intelligence has contributed (formal) design models and software support tools to application areas such as architecture, engineering and software design. This paper explores the effectiveness of applying design models to the area of organization (re)design. To that purpose a component-based model for (re)design of organizations is presented as a specialization of an existing generic design model. Using recently developed formalizations within Organization Theory organization models are described as design object descriptions, and organization goals as design requirements. A formal design process description is presented that models the redesign process for an organization that adapts to changes in the environment. The formally specified and implemented approach to organization redesign thus obtained has been tested for a well-known historical case study from the Organization Theory literature.

Keywords Organizational redesign · Organizational change · Multi-agent systems · Generic architecture

1 Introduction

Organizations are created to smoothen processes in all aspects of society, even in the artificial societies of software agents. From a design perspective organizations

M. Hoogendoorn (✉) · J. Treur
Department of Artificial Intelligence, Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
e-mail: mhoogen@cs.vu.nl

J. Treur
e-mail: treur@cs.vu.nl

C.M. Jonker
EEMCS, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: C.M.Jonker@tudelft.nl

have goals to be achieved or maintained that serve as requirements for their functioning. The behavior of the elements or parts of the organization and their interaction together should result in overall organization behavior that fulfills the goals of the organization. Environmental circumstances impose constraints on the organization with respect to the way its goals can be fulfilled. As the environment changes over time, so do these constraints. To adapt to such changes in constraints, the organization might have to change itself. From a design perspective the changing constraints can be interpreted as changing requirements to a re-organization problem.

Such a change in an organization is not a simple matter, research has shown that over 70 percent of the change programs in human organizations do not achieve the intended goal (Hall et al. 1993; Bashein et al. 1994). Hence, a lot of improvements can be made to increase the effectiveness of organizational changes, whereby one crucial factor is the selection of the most appropriate new organization to change to.

Both within the area of computational model for organizational design and Artificial Intelligence, simulation-based approaches have been proposed that allow for the study of the effectiveness of an organization given particular circumstances, e.g. SimVision (Kunz et al. 1998; Jin and Levitt 1996), OrgAhead (Carley and Svoboda 1996; Carley and Lee 2004), Blanche (Hyatt et al. 1997), and PluralSoar (Carley et al. 1992). Such simulations can give an indication what type of organization can be suitable given a certain type of requirement imposed on the organization. Also within Contingency Theory (see e.g. Donaldson 2001) such information is expressed, but on a more high-level.

Within the area of AI and Design, in the last decade formally specified generic models for (re)design processes have been developed; e.g., Bosse et al. (2010), Brazier et al. (1998). The redesign process in Brazier et al. (1998) for example involves generation and modification steps for the specification of the requirement set and for the design object description. In order to apply such a redesign model in the area of organizations requires the instantiation of the model with specialized knowledge on: (1) organization goals; (2) how to derive refined requirements from such goals given a variable environment; (3) the current design object description, and (4) what components for a design object satisfy which requirements. A redesign process results in a new design object description as a modification of the existing one and a specification of changed (new) design requirements, and hence, can give the change manager good support in deciding when change is needed, and what change would be most effective to perform.

In this paper, a first application of the techniques from the domain of AI and Design within the area of organizational modeling is presented. Hereby, a model is presented that continuously monitors the requirements of the organization (e.g. making a profit), and is triggered in case these requirements are no longer fulfilled. A process is then started which eventually generates the most appropriate new form of an organization (based upon organizational templates that are available, for example as a result of the aforementioned simulators in the domain of Computational and Mathematical Organization Theory) given the current environment. In order to make such an application possible, formalized organizational models (see e.g. Ciancarini and Wooldridge 2001; Hannoun et al. 1998, 2000; Hubner et al. 2002; Jonker and Treur 2003) are used as the objects to be designed, and for the requirements of these models formalizations of organizational behavior are used (see e.g. Hannoun et al. 1998,

2000; Hubner et al. 2002; Jonker and Treur 2003). The whole design process has been formalized by means of a temporal logical language (cf. Bosse et al. 2010). The main contribution of this paper is to present a generic, reusable, architecture for support in organizational change, and to show that this model can indeed work for a particular case (in this case the lean production versus mass production, inspired on the Ford/Toyota case as described in Womack et al. 1991). In order to show the success of the model, formal verification techniques have been used.

This paper is organized as follows. Section 2 presents related work within the domain of Artificial Intelligence and Organization Design. In Sect. 3 the component based model for the design and redesign process is presented and the types of domain specific knowledge needed in such a process is described. Section 4 addresses the formalization of design object descriptions by means of an organization model format in which different components and aggregation levels can be distinguished. In Sect. 5 the relation between goals, a changing environment and requirements is described, including example cases described in Organization Theory. Section 6 presents the method of refinement of such requirements and shows a specific example. Thereafter, Sect. 7 presents examples of design object that are known to satisfy certain design requirements, and Sect. 8 presents generic properties which enable an evaluation of the successfulness of the whole (re)design process. Section 9 presents simulation results of the model whereas Sect. 10 verification of these simulation results is addressed. Finally Sect. 11 is a discussion.

2 Related work

Within the field of computational organization modeling, a variety of approaches to study organizations have been introduced. An extensive argumentation on the usefulness and purpose of computational simulations of organizations has been detailed by Harrison et al. (2007). According to the paper, there are quite some purposes for which a simulation approach with respect to organizations can be used. First of all, they adopt three purposes as identified by Axelrod (1997):

1. **Prediction**: use the simulations to predict what will happen in particular situations, and possibly empirically test these findings.
2. **Proof**: Show that a simulation is able to produce certain types of behavior.
3. **Discovery**: Simulations can also be utilized to identify certain emergent or unanticipated consequences of simple processes.

In addition to the aforementioned reasons as taken from Axelrod, they identify four additional reasons for the use of organizational simulations:

4. **Explanation**: Use simulations to explain particular behaviors within an organization.
5. **Critique**: Examine existing theories about organizations and try to find simpler options that are able to explain the phenomena as well.
6. **Prescription**: Use simulations to show the suitability of a particular method of organizing.

7. **Empirical guidance**: The use of simulations might actually result in new empirical strategies for organizations.

According to Cohen and Cyert (1965) four categories can be defined within computational models of organization behavior:

1. **Descriptive simulation studies**: The main purpose of this type of study is to formulate new theories, test them, and use these theories for future prediction.
2. **Illustrative simulations of quasi-realistic organizations**: These can be used to explore how organizations behave given a certain set of (reasonable) assumptions.
3. **Normative simulation studies for designing organizations**: Investigate which form or organization is best given a certain goal.
4. **Man-machine simulations**: Are meant to train people in order to let them function better in an organization.

Ashworth and Carley (2007) give an extensive overview of the currently existing computational models for organizations. In their article, they have identified twenty-nine existing models that focus on human organizations and networks, incorporate behavior at the level of individuals, and enable the investigation of multiple organizational aspects (i.e. multiple aspects of individual, group, and inter-group behavior). In order to get a good understanding of the existing approaches within organizational modeling and simulation, an overview is presented of a number of these approaches. The overview is by no means meant to be as exhaustive as the overview presented by Ashworth and Carley, but is meant to place the current model in an appropriate context. Hereby, a division is made between the approaches within the domain of Artificial Intelligence, and the approaches specifically designed in the area of Organizational Design. To enable a comparison (presented in the last subsection), the following aspects of the approaches will be briefly described:

- Category of the model (cf. Cohen and Cyert 1965)
- The way in which the actors in the organization are described
- The way in which the tasks within the organization are described
- The way in which the organizational structure is described
- The performance indicators used within the model
- The way in which organizational change is addressed

In the same section, the main incentive of the approach presented in this paper is also compared with the other approaches. For a more extensive comparison, the reader is referred to Ashworth and Carley (2007).

2.1 Organizational design

Within the area of computational models for organizational design, a number of approaches will be discussed, namely *OrgAhead*, *SimVision* (or *VDT*), and *Blanche*. Each of them is discussed in more detail below. A comparison based upon the criteria mentioned above is discussed in Sect. 2.3.

OrgAhead OrgAhead (Carley and Svoboda 1996; Carley and Lee 2004) is a model to study organizational learning and decision making. The idea is that agents have to

solve particular tasks within the organization, and learn how to perform these tasks over time. In addition, the organization itself can be changed to increase the overall effectiveness. The effectiveness of the organization is defined as the percentage of correctly performed tasks. This can be accomplished by: (1) changing the turnover (hire, fire, or replace a person), (2) reassigning tasks, and (3) re-assigning personnel. The organizational changes are initiated after every x tasks that have been performed, and the precise changes are defined using either *hill-climbing* or *simulated annealing*. Before actually changing the organization, the organization is tested to see whether this new structure indeed improves the performance of the organization. The tasks themselves are represented by means of series of binary bits (although trinary bits are also possible).

SimVision SimVision (see e.g. Kunz et al. 1998; Jin and Levitt 1996), also called the *Virtual Design Team*, is meant to study project organizations and investigate how interdependencies between activities impose requirements on coordination within an organization. It can be used to investigate how the design of an organization and the communication tools change the overall performance, and the coordination capacity of the organization or team. In the model, an information processing point of view is taken whereby the actors in the organization are information processing units, and they send and receive messages along specific lines of communication (whereby the communication lines are limited by the organizational structure). A distinction is made between the production work (actual performance of the task) and the coordination work (the communication needed to perform the task), the sum of the two is the total work volume. Tasks in *SimVision* are referred to as high-level tasks (e.g. a task the organization as a whole should perform), activities are the lower level equivalent that can be performed by an individual actor. Each activity is characterized by a work volume, the skill requirements to perform the activity, the complexity of the activity, and the uncertainty a successful performance of the activity. The overall evaluation of the approach is performed based upon the project duration, direct cost, and the coordination quality. Different organizational structures can be tested to investigate their effectiveness. Hereby, the organizational structure essentially expresses the communication lines that are present, the level in the organization where decisions are made, the formalization of the organization (how formal do actors interact with each other), and the likelihood thereof (what is the probability that actors follow the formal or informal interaction lines).

Blanche Blanche (Hyatt et al. 1997) has been designed to study organizational networks, and has been expressed on a very generic level. It essentially describes an organization by means of nodes and links between these nodes. The nodes are the actors within the organization, and can be described by a number of attributes, whereby each attribute is represented by a real number. The links represent channels between the actors, and can represent communication, influence, workflow, activation, or another relationship that might be of interest for the investigation. The relationships between the various concepts that are defined for the nodes and the links can be defined by means of a set of differential equations.

2.2 Artificial intelligence

In general, in the research concerning Multi-Agent Organizations in the domain of Artificial Intelligence, change is not explicitly addressed. Organizations are commonly expressed in a formal fashion by means of a structure of an organization, and a certain description of desired behavior of the organization. They typically abstract from the specific behavior of agents. Some approaches however, do explicitly represent change, for instance, MOISE+ (Hubner et al. 2004), MOISE+ will be explained in more detail below. Also, some approaches study the influence of the agent capacities upon the overall success of the organization, such as PluralSoar (Carley et al. 1992), this approach will also be discussed.

Moise+ Within Moise+ (Hubner et al. 2004) an approach is proposed towards redesign of organizations. In the approach, four phases are identified, namely: (1) monitoring (identify when reorganization is needed), (2) design (how to build a new organization), (3) selection (what organization to move to), and (4) implementation (how to move to the new organization). Reorganization can take place in a pre-defined form (i.e. a plan is already in place and the time at which the reorganization takes place is already known), or a centralized or bottom up form in which the moment of change is not known up front. The organization itself is modeled from both a *structural* as well as a *functional* perspective. In the structural perspective, the organization is characterized by roles, relationships between roles, and groups. The functional specification involves missions (a set of goals) and a global plan (a structure of the goals). The two views are also combined by combining roles with missions using permissions and obligation of roles towards these missions. A software environment is available that allows the expression and enforcement of the organizational specification.

Plural-Soar Within Plural-Soar (Carley et al. 1992) intelligent agents that are based on the Soar architecture are created, and grouped within a small organization. The idea is to study organizations on a micro-level, thereby focusing on the relationship between the skills of the individuals within the organization, the job requirements, and the schemes for coordination. The main task they focus on within their paper concerns the so-called *Warehouse task* in which agent must locate items accompanying a particular order in the warehouse. The actors or agents in the organization should fulfill six basic capabilities in order to be considered sufficiently intelligent: (1) they are able to perceive their environment and take action, (2) they have a memory, (3) they are able to follow certain instructions, (4) they are able to analyze the task at hand and determine the cause of action, (5) they are able to communicate with other agents, and (6) they are able to analyze their social environment. The soar-based agent used in the approach indeed exhibits all these capabilities. The organization structure of the approach currently involves the number of agents, and the communication lines. Each agent is assumed to be independent and can take its own decisions. The evaluation of the success of an organization is based upon the time needed to complete a task, the amount of effort needed (both cognitive and physical), and the process of performing the tasks.

2.3 Comparison

Table 1 provides an overview of the various approaches based upon the criteria mentioned at the beginning of Sect. 2.

The approach presented in this paper (shown in the last row of Table 1) attempts to allow for a formalization of the process of looking for an appropriate organization to change to. Hereby, a certain set of possible organizational structures and behaviors is assumed to be known in advance, and it is also known what the characteristics of these organizational elements are (e.g. what the cost are). This paper then shows the process of monitoring the current organization, and trying to identify whether the organization still meets the requirements. If the organization does not meet the requirements, the approach can find a new organization (based upon the building blocks) to move to that does fulfill the requirements. Such building blocks can be inserted based upon the experiences obtained in the aforementioned simulation environments (OrgAhead, SimVision, Blanche, Plural-Soar), so these should be seen as input for the process indicated here. It can also be seen as a more detailed expression of the change process as expressed in Moise+, in which the process itself is specified on a highly abstract level.

3 A component-based model for (re)design of organizations

This section presents a component-based generic model for design of organizations based on requirements manipulation and design object description manipulation. The component-based model presented draws inspiration from Brazier et al. (1998) and was specified within the DESIRE (Brazier et al. 2002) framework. The model for design is composed of three components, see Fig. 1:

- RQSM, which stands for Requirement Qualification Set Manipulation. Such requirements of the organization are for example acquired by elicitation in cooperation with managers within a company. Within RQSM the appropriate requirements are determined in relation to the goals set for the organization and the current environmental conditions. After having selected a set of requirements, these are refined to more specific ones.
- DODM, for Design Object Description Manipulation, creates a design object description based on the (specific) requirements received from RQSM. In order to determine such a design object description, a number of alternative solutions known to satisfy the requirements are generated and according to certain strategic knowledge one of those is selected.
- Design Process Coordination (DPC) is the coordinating component for the design process. The component determines the global design strategy (e.g., Brazier et al. 1998) and can evaluate whether the design process is proceeding according to plan.

Information exchange possibilities are represented by the links between input and output of the components and the input and output of the model. Input and output are represented by the small boxes left and right of components.

The next sections describe the three components in more detail. The model as described here, is a generic design model for organizational design without application-

Table 1 Comparison of existing organization modeling approaches

Model	Main category of the model (cf. Cohen and Cyert 1965)	Actor description	Task description	Organization structure description	Performance indicators	Organizational change modeling
OrgAhead	Descriptive simulation studies	By means of a decision rule and maximum resources	Tasks are represented as binary or trinary type bits that need to be classified	Multi-level hierarchy	Accuracy of performance	– Actors in organization learn – Reassign tasks – Hire/fire/replace actor – Reassign personnel
SimVision	Descriptive simulation studies	As information processing entities with certain skills	Tasks are represented by the work volume, skill requirements, complexity, and the uncertainty	By communication lines, the amount of centralization, and formalization	– Project duration – Direct cost – Coordination quality	– Allows for investigation of different organizations to study appropriate change, no explicit modeling of change
Blanche	Descriptive simulation studies	As nodes in a graph with certain attributes	No explicit representation, can be done using the attributes of the actors and links	By means of a graph structure with actors and links between them.	Depends on the attributes defined within the organization	– No explicit modeling of change, but can be used to investigate different org. types
Moise+	Illustrate simulations of quasi-realistic organizations	By means of role descriptions and obligations and permissions	No explicit task representation, partially doable by means of goals	By means of role, groups, and relationships between them	Depends on the goals set	– Change is modeled using explicit reorganization structure in four phases
Plural-Soar	Descriptive simulation studies/ Normative simulation studies for designing orgs.	As intelligent agents using Soar	Currently the task description involves a simple *Warehouse* task	Number of actors and the communication lines between the actors	– Time to complete task – Effort needed – Process of performing tasks	– No explicit modeling of change, but can be used to investigate different org. types
This paper	Normative studies for designing organizations	From an abstract view: By means of formal behavioral role descriptions	The tasks are expressed by means of formal behavioral role descriptions as well	By means of roles, groups and interactions between those elements	Depends on the properties defined for the organization	Change is addressed as a redesign process, requirements are formulated for change, and modifications in the organizations are made based upon changes that satisfy the requirements

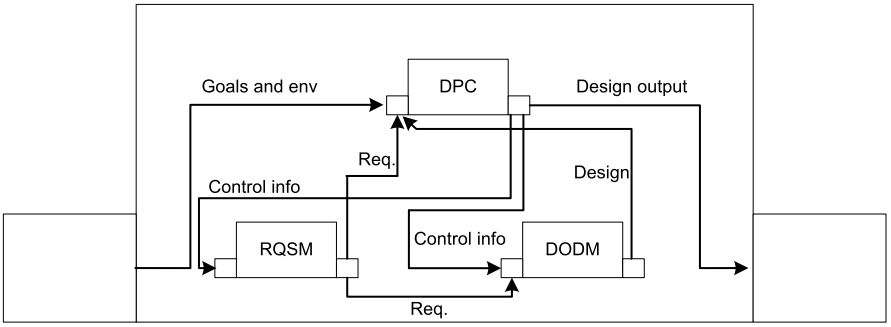


Fig. 1 Top level of the design model

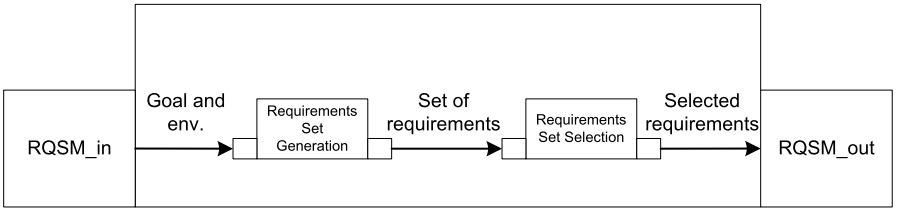
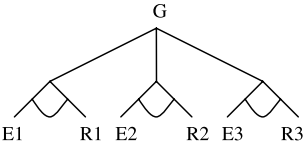


Fig. 2 Components within RQSM

Fig. 3 Example AND/OR tree relating environmental conditions and requirements to a goal



or domain-specific knowledge. In later sections such knowledge is specified for a case study.

3.1 RQSM

The component RQSM is composed from two sub-components, namely Requirements Sets Generation and Requirements Set Selection, see Fig. 2.

The component Requirements Sets Generation receives as an input the current environmental conditions and the organizational goals. The sub-component contains knowledge on what requirements entail fulfillment of organizational goals given the environmental conditions. Such knowledge can be depicted in the form of AND/OR trees as shown in Fig. 3.

If for example E1 is observed, requirement R1 is an example of a requirement that, when fulfilled, guarantees to satisfy goal G under environmental conditions E1. If the environment changes to situation E2, the requirement has to change as well; the example tree shows how R1 can be changed to requirement R2 that guarantees G under the new environmental conditions E2. After a requirement is determined,

it can be refined in order to obtain requirements on a more specific level. Making such a requirement more specific can result in several options being generated. For example, it might be possible to establish a certain market share by having the best quality products but also by having the lowest priced products. After having refined each of the requirements, all possible sets of refined requirements are forwarded to the component Requirements Set Selection.

After the component Requirements Set Selection has received the alternative sets of requirements its task is to select one of those alternatives, and to forward it to the component DODM which will in turn find a suitable organization design for such a requirement set. Different selection methods exist, e.g., explicit ranking, on the basis of strategic knowledge. Such strategic knowledge can for example be based on the source of requirements: requirements that originate from users can for example be preferred over those derived by default rules which are in turn preferred over requirements derived from previous requirements (see Haroud et al. 1994).

3.2 DODM

DODM receives a set of refined requirements from RQSM, which is handled by two sub-components, Design Object Description Generation and Design Object Description Selection. The design object descriptions are descriptions of designs of the organization, including both structural aspects as behavioral aspects.

Design Object Description Generation receives the requirements and delivers descriptions of possible alternative design objects (i.e., organization design descriptions), such that the (specific) requirements as received from RQSM are satisfied. To establish satisfaction, knowledge is needed that specifies what part of a design object contributes to fulfillment of a specific requirement. If, for example, the requirement is to produce products of the highest quality, then a satisfactory design is an organization having a department dedicated to checking quality and repairing of production errors. Again, there can be many possibilities available that satisfy the requirements. All alternatives found are forwarded to the component Design Object Description Selection.

The component Design Object Description Selection can use several criteria to choose the optimal design, such as operational costs effectiveness, and production time effectiveness. In order to make such a selection, the component has (strategic) knowledge concerning these aspects. It might for example know the typical price for hiring an agent for a particular role Eventually, the component outputs a new design for the organization.

3.3 DPC

The component DPC is the component which determines the global design strategy and oversees whether the design process proceeds according to plan. Two different tasks are distinguished. DPC checks whether a design object description determined by DODM satisfies the refined requirements. It might for example be the case that the combination of two suitable design object parts causes a conflict. In case the refined requirements are not satisfied control information is passed to DODM stating that an

alternative should be found (e.g., taking a different branch of an OR tree). In case these refined requirements are satisfied whereas the high-level requirements are not, the requirements refining process has failed, therefore control information is given to RQSM to refine the requirements in another way (again by for example taking another OR branch).

4 Organization models as design objects

An organizational structure defines different elements in an organization and relations between them. The dynamics of these different elements can be characterized by sets of dynamic properties. An organizational structure has the aim to keep the overall dynamics of the organization manageable; therefore the structural relations between the different elements within the organizational structure have to impose relationships or dependencies between their dynamics; cf. Jonker and Treur (2003). In the introduction to their book Lomi and Larsen (2001) emphasize the importance of such relationships:

> 'given a set of assumptions about (different forms of) individual behavior, how can the aggregate properties of a system be determined (or predicted) that are generated by the repeated interaction among those individual units?'
> 'given observable regularities in the behavior of a composite system, which rules and procedures—if adopted by the individual units- induce and sustain these regularities?'

Both views and problems require means to express relationships between dynamics of different elements and different levels of aggregation within an organization. In Lomi and Larsen (2001) two levels are mentioned: the level of the organization as a whole versus the level of the units. Also in the development of MOISE (Hannoun et al. 2000; Haroud et al. 1994; Hubner et al. 2002) an emphasis is put on relating dynamics to structure. Within MOISE dynamics is described at the level of units by the goals, actions, plans and resources allocated to roles to obtain the organization's task as a whole. Specification of the task as a whole may involve achieving a final (goal) state, or an ongoing process (maintenance goals) and an associated plan specification.

The approach in this paper is illustrated for the AGR (Ferber and Gutknecht 1998) organization modeling approach. Figure 4 shows an example organization modeled using AGR. Within AGR organization models three aggregation levels are distinguished: (1) the organization as a whole; the highest aggregation level, denoted by the big oval, (2) the level of a group denoted by the middle size ovals, and (3) the level of a role within a group denoted by the smallest ovals. Solid arrows denote transfer between roles within a group; dashed lines denote inter-group interactions. This format is adopted to formalize organization models as design object descriptions. In addition, behavioral properties of elements of an organization are part of a design object description. TTL (Jonker and Treur 2002) is used to express such behavioral properties.

In TTL state ontology is a specification (in order-sorted logic) of a vocabulary. A state for ontology `Ont` is an assignment of truth-values {`true, false`} to the

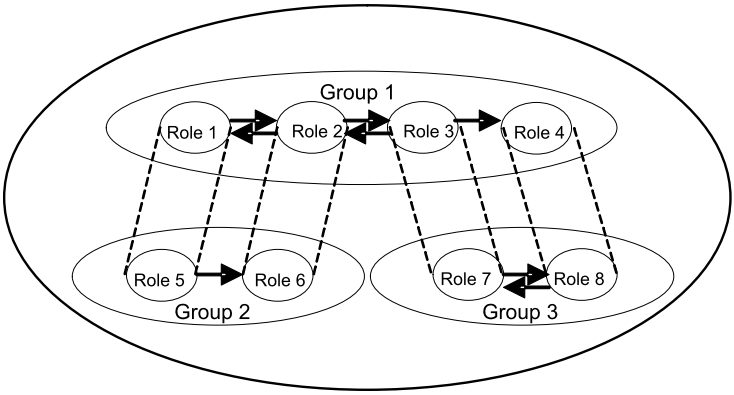


Fig. 4 An AGR organization structure

set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : \mathtt{T} \rightarrow \mathtt{STATES(Ont)}$, i.e., a sequence of states γ_t ($t \in \mathtt{T}$) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP($\sum$) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the state in γ at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation $\models$, comparable to the Holds-predicate in the Situation Calculus: state(γ, t) $\models$ p denotes that state property p holds in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$. A special software environment has been developed for TTL, featuring both a Property Editor for building and editing TTL properties and a Checking Tool that enables formal verification of such properties against a set of (simulated or empirical) traces.

5 RQSM: changing requirements upon environmental change

Organizational requirements change due to changing environmental circumstances. The circumstances are input to RQSM. The general pattern is follows. A certain organizational goal G (e.g. sufficient demand) is no longer reached, due to an environmental change, say from E1 to E2. In the old situation requirement R1 was sufficient to guarantee G under environmental condition E1: E1 & R1 $\Rightarrow$ G. Here R1

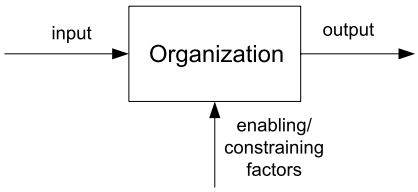


Fig. 5 Flow of information in an organization

is a requirement expressing a relation which states that under the condition E1 the organization is able to achieve G. The change from E1 to E2 makes that requirement R1, which is still fulfilled but has become insufficient, is to be replaced by a new, stronger requirement R2 which expresses that under environment E2 goal G can be achieved; therefore: E2 & R2 $\Rightarrow$ G. Thus, the organization is triggered to change to fulfill R2 and as a consequence fulfill goal G again.

Jaffee (2001) distinguishes several of these external triggers for organizational change. This paper presents a classification (see Fig. 5) of those triggers based on the flow of information for an organization. The input type of external trigger includes the triggers the organization notices on its input, for example changes in the resources or suppliers. *Enabling/constraining factors* are external triggers such as government rules and technology that concern processes within the organization. Finally, *output* can influence the input of an organization and can therefore affect the triggers received by an organization. Output information itself is however not considered a trigger for organizational change.

5.1 Input changes

The input of an organization can originate from a variety of different sources. Each of these sources can cause a change of requirements, and possibly trigger an organization to change.

A first source is formed by the *suppliers* who can increase their price of a product P, which is used by the organization for the production, at time t from M_1 to M_2. A formal form of this environmental condition is specified in E1 using the Temporal Trace Language (TTL) as explained in Sect. 3.

E1(P, M, t): Supplier Price
$\exists$R:REAL state(γ, t) $\models$ environmental_condition(price(P, R), pos) & R $\leq$ M

Before the environmental change, E1(P1, M_1, t) specifies the relevant property of the environment. After the change of supplier price however, this property no longer holds whereas E1(P1, M_2, t) does hold. The overall goal to be maintained within the organization is to keep the demand of product P above a threshold D. A formal specification of the goal is presented in OP1.

OP1(P, D, t): Sufficient demand
$\exists$I:INTEGER
state(γ, t) $\models$ environmental_condition(customer_demand(P, I), pos) & I $\geq$ D

The requirement imposed for the organization is to maintain the goal of keeping demand for product P2 above D, in the new situation given the environmental condition of the price M for product P1 which is needed for the production of P2. This requirement is specified below in property R.

R(P1, P2, M, D): Maintain demand
$\forall$t:TIME
[state(γ, t) $\models$ needed_for_production_of(P1, P2) & E1(P1, M, t)] $\Rightarrow$ OP1(P2, D, t)

Before the change in the environment, requirement R1 which is R(P1, P2, M_1, D) was sufficient to ensure the goal being reached. After the change however, this requirement is still satisfied but might be insufficient to ensure the goal. This is due to the fact that the environmental condition E1 in the antecedent of E1 & R1 $\Rightarrow$ G does not hold, and hence, cannot be used to entail G (although the requirement R1 is fulfilled all the time). The requirement is therefore withdrawn and replaced by the requirement R2 which is R(P1, P2, M_2, D). This R2, however, is not necessarily satisfied and may require an organizational change to enable fulfillment.

Secondly, an input trigger can be formed by *resources* that run out, becoming a lot more expensive. Therefore, the requirement for an organization triggered in such a way is to reduce the usage of the particular resource. This can for example be accomplished by focusing on a completely different, more viable product, or producing the same goods using different resources.

Another source is formed by the *customers* whose demands decreases for the good being produced. The organization can change direction (and thus change the organization) or keep producing the same good but decrease the output (and therefore also change the organization).

Finally, *competitors* might change their production methods causing a more efficient production process for products within the same product group as P, lowering their price from C_1 to C_2.

5.2 Changes in enabling/constraining factors

Besides triggers on the input of an organization, another type of trigger exists: the enabling and constraining factors. First of all, the enabling factors within the organization include *technology*. In case the technology available to produce a product P changes from T1 to T2, the profit margin should remain at least at the same level D for a company.

OP′(P, D, t): Sufficient Profit Margin
$\exists$R:REAL state(γ, t) $\models$ belief(profit_margin(P, R), pos) & R $\geq$ D

E′(P, T, t): New Technology
state(γ, t) $\models$ environmental_condition(technology_available_for(T, P), pos)

R′(P, T, D): Maintain Profit
$\forall$t:TIME E3(P, T, t) $\Rightarrow$ OP1(P, D, t)

All properties have been specified similar to those presented in the previous subsection. Before the environmental change of available technology E′(P, T1, t) was the

case whereas E′(P, T2, t) is the new environment. Secondly, constraining forces include *government regulations and labor aspects*. Government regulations for workers might affect human resource practices and composition of the workforce. Concerning labor aspects, the union might demand a reduction from 40 to 36 hours a week, which naturally causes organizational change. All these aspects should however not decrease overall profitability of the organization.

6 RQSM: refining requirements based on interlevel relations

To fulfill requirements at the level of the organization as a whole as discussed in Sect. 5, parts of the organization need to behave adequately (see also the central challenges put forward by Lomi and Larsen (2001) as discussed in Sect. 4). Based on this idea, in this paper dynamics of an organization are characterized by sets of dynamic properties for the respective elements and aggregation levels of the organization. An important issue is how organizational structure (the design object description determined in DODM) relates to (mathematically defined) relationships between these sets of dynamic properties for the different elements and aggregation levels within an organization (cf. Jonker and Treur 2003). Preferably such relations between sets of dynamic properties would be of a logical nature; this would allow the use of logical methods to analyze, verify and validate organization behavior in relation to organization structure. Indeed, following Jonker and Treur (2003), in the approach presented below, logical relationships between sets of dynamic properties of elements in an organization turn out an adequate manner to (mathematically) express such dynamic cross-element or cross-level relationships.

A general pattern for the dynamics in the organization as a whole in relation to the dynamics in groups is as follows:

```
dynamic properties for the groups &
dynamic properties for inter-group interaction
-> dynamic properties for the organization
```

Moreover, dynamic properties of groups can be related to dynamic properties of roles as follows:

```
dynamic properties for roles &
dynamic properties for transfer between roles
-> dynamic properties for a group
```

The idea is that these are properties dynamically relating a number of roles within one group.

A generic overview of the logical relationships between dynamic properties at different aggregation levels is depicted as an AND-tree in Fig. 6. It is possible that each level shown in the tree (for example organization properties) again consists of multiple levels. The logical relationships put forward above can be formalized further as shown in Jonker and Treur (2003).

Figure 7 shows an example of a hierarchy of dynamic properties for an organization producing certain products, the properties follow field observations at the Ford

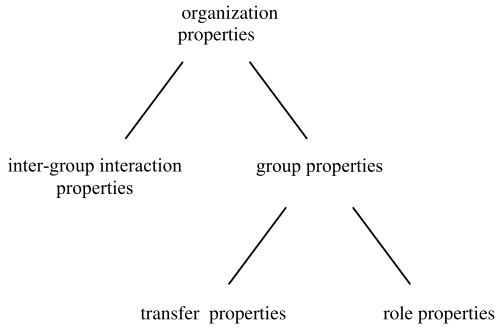


Fig. 6 Overview of relations between dynamic properties

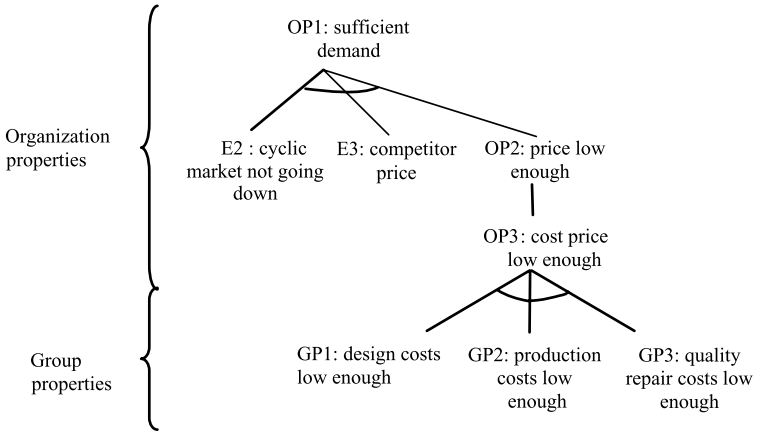


Fig. 7 Hierarchy of organizational and group properties

Motor Company in 1980 described in Womack et al. (1991). This example is used as an illustration and a case study for the approach put forward in this paper. The overall organizational goal is to maintain sufficient demand for the goods being produced, as was also the case in OP1 in Sect. 5. The organization has separate departments for design, production and quality control, which are modeled as groups in the organization. The highest levels represent organizational properties or goals at the aggregation level of the organization as a whole, whereas the lowest level shown here represents properties at the aggregation level of the groups. Note that the fact that these are group properties already restricts the design of the object in DODM, which makes the process less complex.

A definition for each of the properties in Fig. 7 is presented below. Notice that this hierarchy could easily be extended by other aspects (e.g., of quality of the products as a reason for the demand decreasing or not).

Property OP1 is described in Sect. 5. One of the environmental conditions is that the cyclic market is not going down for a product P at time t in case the demand for the product group as a whole (i.e., all goods produced by different companies in this particular category) is not going down.

E2(P, t): Cyclic market not going down
∀G : PRODUCT_GROUP, I1, I2:INTEGER
[state(γ, t) ⊨ belongs_to_product_group(P, G) &
state(γ, (t-1)) ⊨ environmental_condition(customer_demand(G, I1), pos) &
state(γ, t) ⊨ environmental_condition(customer_demand(G, I2), pos)]
⇒ I2 ≥ I1

Furthermore, an environmental condition E3 poses a requirement on the price of competitors in the form of the average price of products within the product group to which product P belongs. These prices should not be higher than V:

E3(P, V, t): Competitor Price
∀G : PRODUCT_GROUP, V1 : REAL
[state(γ, t) ⊨ belongs_to_product_group(P, G) &
state(γ, t) ⊨ environmental_condition(average_price(G, V1), pos) & V1 ≥ V]

To achieve goal OP1 given environmental conditions E2 and E3, the price of the products being produced by the organization should be low enough, which in turn is the requirement posed on the organization. Prices are considered low enough for a product P at time t in case the price for the product is equal or below the average price level within the product group (i.e. prices are ≤ V as set above).

OP2(P, V, t): Price low enough
∀V1 : REAL [state(γ, t) ⊨ price(P, V1)] ⇒ V1 ≤ V

Whether the price is low enough depends on the cost price for the particular product P at time t, which purely depends on the costs for the different groups within the organization, as expressed in the group properties (GP's).

OP3(P, V, t): Cost price low enough
∀ V1, V2, V3 : REAL
[state(γ, t) ⊨ design_cost(P, V1) &
state(γ, t) ⊨ production_cost(P, V2) &
state(γ, t) ⊨ quality_repair_cost(P, V3)]
⇒ V1 + V2 + V3 ≤ V

Finally, the individual group properties can be specified such that the costs of each group are below a certain value. that the division of such costs over groups is a refinement choice. An example decision could be the to allow only a small percentage of the costs for quality repair and to divide the brunt of the costs equally over production and design. Each group should meet their individual requirements. First of all, design costs should be low enough:

GP1(P, V1, t): Design costs low enough
∀Q : REAL [state(γ, t) ⊨ design_cost(P, Q)] ⇒ Q ≤ V1

Also, the production costs for product P should be low enough:

GP2(P, V2, t): Production costs low enough
∀Q : REAL [state(γ, t) ⊨ production_cost(P, Q)] ⇒ Q ≤ V2

Finally, quality repair costs should be low enough for product P:

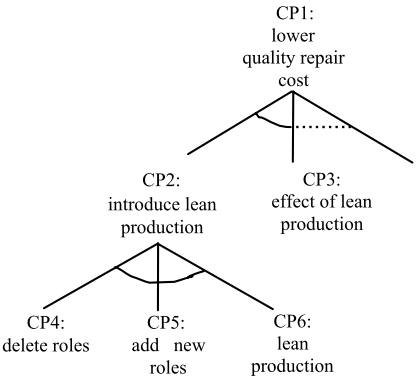


Fig. 8 Redesign options specified in the form of an AND/OR tree

GP3(P, V3, t): Quality repair costs low enough
$\forall Q{:}\mathrm{REAL}\,[\mathtt{state}(\gamma, t) \models \mathtt{quality\_repair\_cost}(P, Q)] \Rightarrow Q \leq V3$

After having generated all options in RQSM, selection knowledge is used to select one of the available options. In this paper, such selection knowledge is not further addressed. The output of RQSM is, however, of the form `selected_basic_refinement_set(RS)` where RS is a name for a requirements set. The elements within this set are defined as follows: `in_selected_basic_refinement_set(R, RS)` where R is a requirement, as the ones shown above, and RS is the selected basic refinement set.

7 DODM: constructing design objects

As stated in Sect. 3, DODM contains a library of templates for (parts of) design objects which are known to satisfy certain requirements (of the form as specified in the last paragraph of the previous section). For the case study used in this paper, the DODM library contains two templates. One of those is a template in which a mass production system is used to produce goods. Such a system produces goods at reasonable production costs but at high quality repair costs. The template for mass production includes a group of production workers (e.g. a production worker for attaching a wheel to a car). The mass production template also contains a quality repair department of considerable size with quality repair worker roles.

The second template in the library is a lean production organization. Lean production has no quality repair costs, since there is no separate quality repair department. The production costs are at the same level as the production costs for mass production organizations. In the lean production method (see e.g. Womack et al. 1991), multitask production workers are present which perform several tasks, and also handle errors in case they are observed. As a result of such immediate error detection and correction, a quality repair department is not present within a lean production model.

Figure 8 shows an example AND/OR tree for DODM (focusing at lean production as a solution) in which options for changes in a design object not satisfying the requirement that design costs are low enough. The specific changes in the design object

are presented below. First of all, the highest level property states that design costs will at least at the required level within a duration d:

CP1(P, D, t): Lower Quality Repair Costs
∀V1, V2 : REAL
[state(γ, t) ⊨ selected_basic_requirement_in (GP3(P, V1, t), RS) &
state(γ, t) ⊨ DOD_includes(D, quality_repair_cost(P, V2)) & V1 < V2]
⇒ ∃t2 : TIME > t, V3:REAL
[t2 < t + d & state(γ, t2) ⊨ DOD_includes (D, quality_repair_cost(P, V3)) & V3 ≤ V1]

On a lower level, property CP2(P, D, t) specifies the introduction of lean production into an organization. This reduces the quality repair costs to 0 as shown by CP3(P, D, t). Although more options are possible for reducing quality repair costs, shown by the dots in Fig. 8, these are not addressed in this paper.

CP2(P, D, t): Introduce Lean Production
∀V1, V2:REAL
[state(γ, t) ⊨ selected_basic_requirement_in(GP3(P, V1, t), RS) &
state(γ, t) ⊨ DOD_includes(D, quality_repair_cost(P, V2)) & V1 < V2]
⇒ ∃t2 : TIME > t
[t2 < t + d & state(γ, t2) ⊨ DOD_includes(D, lean_production_method(P))]

CP3(P, D, t): Effect of Lean Production
[state(γ, t) ⊨ DOD_includes(D, lean_production_method(P))
⇒ state(γ, t) ⊨ DOD_includes(D, quality_repair_cost(P, 0))]

Introducing a lean production system entails that within the production process the specialized roles for mass-production and quality repair department are deleted.

CP4(P, D, t): Delete Roles
[state(γ, t) ⊨ DOD_includes(D, lean_production_method(P))
∃t2:TIME > t
[t2 < t + d &
state(γ, t2) ⊨ ¬DOD_includes(D, exists_role(spec_production_worker)) &
state(γ, t2) ⊨ ¬DOD_includes(D, exists_group(quality_repair_group))]]

Moreover, roles are created that perform multiple tasks, and teams are created such that the roles combined in the team have all the abilities to make a car.

CP5(P, D, t): Add New Roles
[state(γ, t) ⊨ DOD_includes(D, lean_production_method(P))
∃t2:TIME > t, ∀A : AGENT, R : ROLE
[t2 < t + d &
state(γ, t2) ⊨ DOD_includes(D, exists_role(multi_task_production_worker)) &
state(γ, t2) ⊨ DOD_includes(D, previously_allocated_to(A, R, quality_repair)) &
state(γ, t2) ⊨ DOD_includes(D, allocated_to(A, multi_task_production_worker, production_group))]]

Agents that were allocated to the deleted roles in the production process are allocated to the newly formed roles. Agents formerly allocated to a role in quality repair are

fired. Once the system is organized in this fashion, quality repair in a separate department becomes obsolete, and quality repair costs are down to 0 as the production workers are now performing the task. CP6 expresses that the measures as described in CP4 and CP5 results in a lean production method for the product P:

CP6(P, D, t): Lean Production
∀A:AGENT, R:ROLE
[state(γ, t) ⊨ ¬ DOD_includes(D, exists_role(spec_production_worker)) &
state(γ, t) ⊨ ¬ DOD_includes(D, exists_group(quality_repair_group)) &
state(γ, t) ⊨ DOD_includes(D, exists_role(multi_task_production_worker)) &
state(γ, t) ⊨ DOD_includes(D, previously_allocated_to(A, R, quality_repair))
state(γ, t) ⊨ DOD_includes(D, allocated_to(A, multi_task_production_worker, production_group))]
⇒
∃t2:TIME < t + d
state(γ, t2) ⊨ DOD_includes(D, lean_production_method(P))

After such options for (re)design of the object have been generated based on the requirements, selection knowledge is used to select one of the options that have been generated. This knowledge is not addressed in this paper. Eventually, DODM outputs a design object description of the form `selected_DOD_output(D)` where D is the design object description. Furthermore to identify properties of the DOD or its parts, output of the form `in_selected_DOD_output(P, D)` is generated where P is a property of (a part of) the DOD and D is the selected DOD. This is based on the internal information represented in the form of `DOD_includes(D, P)`.

8 (Re)design process evaluation

This section addresses the evaluation of the whole design process. The overall design process is successful when both RQSM and DODM show the proper behavior.

RQSM shows the proper behavior in case it generates requirements, and these requirements indeed result in the goal set for the organization being met. Such properties are formulated in a formal form below.

RQSM_generate
If RQSM receives new environmental conditions on its input, then RQSM eventually generates a set of requirements.

∀t:TIME, γ: TRACE, E:ENV_COND
state(γ, t, input(RQSM)) ⊨ environment_property(E) &
¬∃t′:TIME < t [state(γ, t′, input(RQSM)) ⊨ environment_property(E)]
⇒ ∃t2:TIME > t, G:GOAL, RS:REQUIREMENT_SET
[state(γ, t2, output(RQSM)) ⊨ main_requirement(G) &
state(γ, t2, output(RQSM)) ⊨ selected_basic_refinement_set(RS)]

RQSM_successful
If RQSM generates requirements, then the combination of these requirements entail the goal set for the organization.

∀t:TIME, γ:TRACE, RS:REQUIREMENT_SET, G:GOAL
[state(γ, t, output(RQSM)) ⊨ main_requirement(G) &
state(γ, t, output(RQSM)) ⊨ selected_basic_refinement_set(RS)]
⇒ entails_goal(RS, G)

DODM shows the proper behavior in case it first of all generates a design object description in case a new requirement set is received. Besides simply generating such a design object description, the object also needs to satisfy the requirements received on its input.

DODM_generate
If DODM receives a new requirements set on its input, then DODM eventually generates a design object description as output.

$\forall$t:TIME, γ:TRACE, RS:REQUIREMENTS_SET
[state(γ, t, input(DODM)) $\models$ selected_basic_refinement_set(RS) &
$\neg\exists$t′:TIME < t
  state(γ, t′, input(DODM)) $\models$ selected_basic_refinement_set(RS)]
$\Rightarrow \exists$t2:TIME, D:DESIGN_OBJECT_DESCRIPTION
  state(γ, t2, output(DODM)) $\models$ selected_DOD_output(D)]

DODM_successful
If DODM generates a design object description as output, then the design object description satisfies the requirements set on the input of DODM.

$\forall$t:TIME, γ:TRACE, R:REQUIREMENT_SET,
D:DESIGN_OBJECT_DESCRIPTION
[state(γ, t, input(DODM)) $\models$ selected_basic_refinement_set(R) &
state(γ, t, output(DODM)) $\models$ selected_DOD_output(D)]
$\Rightarrow$ fulfills_requirements(D, R)

9 Simulation results

In order to show that the model is indeed able to select an appropriate organizational change based upon the formal rules discussed above, a case study has been conducted. The results for one of these simulation runs are presented in this section. Note that these simulations are meant as a proof of concept. The simulation has been performed using a subset of the Temporal Trace Language (TTL) called *leads to*. This is an executable format that can be used to obtain a specification of a simulation model in terms of local dynamic properties (the leaves of the tree in Fig. 6). The format is defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

if	state property α holds for a certain time interval with duration g,
then	after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see Bosse et al. (2007). A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

The setup of the simulation is as follows: A historic case taken from Womack et al. (1991) is used as an input for the model. The case concerns the Ford Motor Company who has been one of the leading car manufacturers since the introduction of mass

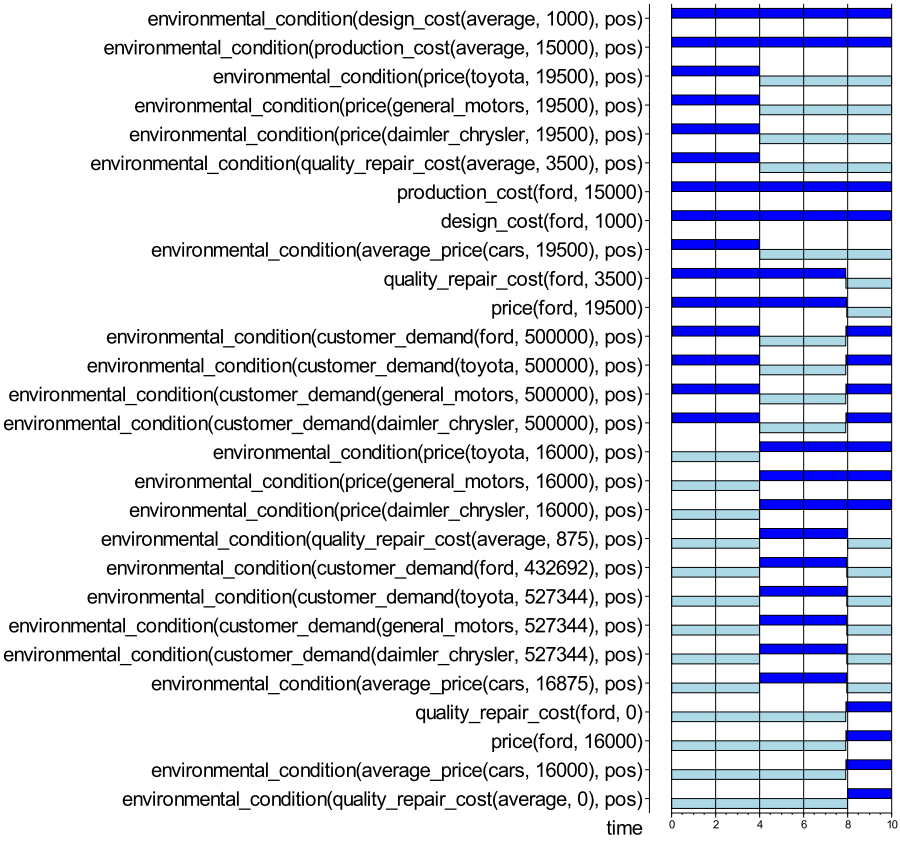


Fig. 9 High-level simulation results using the redesign model

production in 1913. In 1980 the Ford Motor Company suffered a major crisis. The company began to loose vast amounts of money a vast amount of car demand. The model presented in this paper is used to reorganize the Ford organization such that demand is restored again.

9.1 Simulation results: high-level overview

First of all, results are presented in this section that abstract from the details of the organization and the internal functioning of the model. This is to show that on this high level the model indeed shows the expected results. In the following sections, more details will be shown regarding the internal functioning of the model.

The results of the simulation in the form of a trace are shown in Fig. 9. In the figure, the left side shows the relevant atoms, the right part represents a time-line indicating when an atom is true (dark box) or false (lighter box). It can be observed in the figure that initially the market conditions are equal for the four car manufacturers included in the simulation. First of all, the average costs for design, production, and quality repair are the same:

```
environmental_condition(design_cost(average, 1000), pos)
design_cost(ford, 1000)
environmental_condition(production_cost(average, 15000), pos)
design_cost(ford, 15000)
environmental_condition(quality_repair_cost(average, 3500), pos)
quality_repair_cost(ford, 3500)
```

As a result, the price for these cars is the same, also resulting in the same demand for cars from the four manufacturers (it is assumed here that there is no preference of customers for particular brands, if the price is the same, each manufacturer gets an equal share of the total demand).

```
environmental_condition(customer_demand(ford, 500000), pos)
environmental_condition(customer_demand(general_motors, 500000), pos)
environmental_condition(customer_demand(toyota, 500000), pos)
environmental_condition(customer_demand(daimler_chrysler, 500000), pos)
```

Suddenly however, at time point 4 the other three manufacturers lower their price whereas Ford does not:

```
environmental_condition(price(general_motors, 16000), pos)
environmental_condition(price(toyota, 16000), pos)
environmental_condition(price(daimler_chrysler, 16000), pos)
```

This lowering of the price is performed due to a drop in the cost for quality repair cost of the other companies:

```
environmental_condition(quality_repair_cost(average, 875), pos)
```

As a result, demand for Ford cars drops whereas the other manufacturers see an increase in demand:

```
environmental_condition(customer_demand(ford, 432692), pos)
environmental_condition(customer_demand(general_motors, 527344), pos)
environmental_condition(customer_demand(toyota, 527344), pos)
environmental_condition(customer_demand(daimler_chrysler, 527344), pos)
```

Now the model introduced in this paper comes into play. The results obtained after application of this model are shown in the figure as well, using the new organization structure brings the quality repair cost of Ford down to 0 as well:

```
quality_repair_cost(ford, 0)
```

As a result, demand is restored again to the old value of 500,000 cars. These results indeed correspond to the results described in the historic case.

9.2 RQSM simulation results

In order to achieve the result of restoring demand for Ford cars, RSQM and DODM are used to redesign the organization of Ford. In this section, RQSM is addressed.

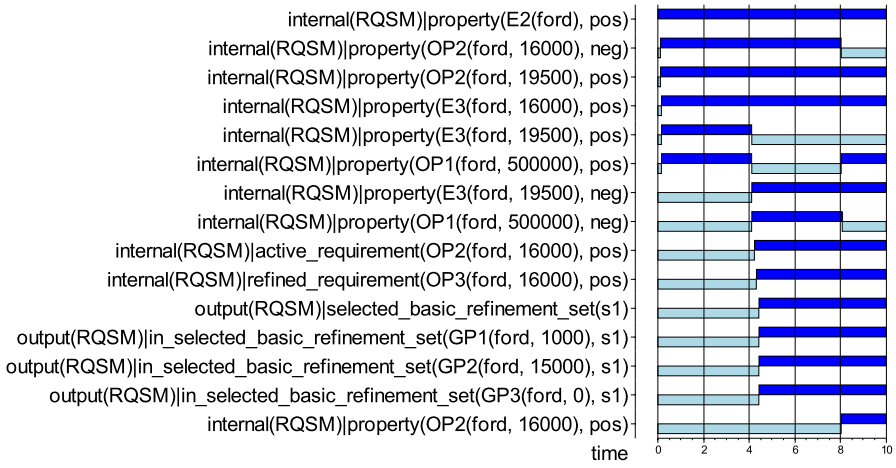


Fig. 10 RQSM reasoning process

Figure 10 shows the atoms related to the RQSM component. As input, RQSM receives the environmental conditions as shown in the trace of the previous section. The goal of the organization is set to keep demand above or at least equal to 500,000 cars (a quarter of the constant total demand for cars of 2,000,000), which is initially satisfied:

```
internal(RQSM)|property(OP1(ford, 500000), pos)
```

The environmental conditions under which the initial Ford organization is obtaining its goal are the following:

```
internal(RQSM)|property(E2(ford), pos)
internal(RQSM)|property(E3(ford, 19500), pos)
```

Which means that first of all, the cyclic market is not going down, and secondly, that the competitor prices are not below 19,500. Given these environmental conditions, OP2(ford, 19500) is indeed a sufficient requirement posed upon the organization to guarantee satisfaction of the overall goal. From time point 4 and on however, the environmental condition E3(ford, 19500) no longer holds due to competitors lowering their price. Another condition does however hold:

```
internal(RQSM)|property(E3(ford, 16000), pos)
```

Given this new environmental condition, property OP2(ford, 19500) is no longer sufficient to obtain the goal:

```
internal(RQSM)|property(OP1(ford, 500000), neg)
```

A new requirement is determined by RQSM that will satisfy the goal under these new environmental conditions:

```
internal(RQSM)|active_requirement(OP2(ford, 16000), pos)
```

This requirement is thereafter refined until the level of basic requirements of which a set is sent to the output:

```
output(RQSM)| selected_basic_requirement(s1)
output(RQSM)|in_selected_basic_refinement_set(GP1(ford, 1000), s1)
output(RQSM)|in_selected_basic_refinement_set(GP2(ford, 15000), s1)
output(RQSM)|in_selected_basic_refinement_set(GP3(ford, 0), s1)
```

In this case the selected basic refinement includes bringing down the cost of the quality repair cost to 0 whereas the requirements for the rest of the costs (i.e. production and design) remain the same.

9.3 DODM simulation results

Figure 11 shows the simulation results for the DODM component. After RQSM has refined and outputted these requirements, DODM receives these on its input. Furthermore, DODM has knowledge about the current organization used by the Ford organization:

```
DOD_includes(ford_design, exists_group(design_group), pos)
DOD_includes(ford_design, exists_group(production_group), pos)
DOD_includes(ford_design, exists_group(quality_repair_group), pos)
DOD_includes(ford_design, role_belongs_to_group(spec_prod_worker,
              production_group), pos)
```

The Ford organization consists of three groups, namely a design group, a production group, and a quality repair group. Furthermore, the production group consists of specialized production workers. In other words, Ford is using a mass production type of company. After having received the basic refinement set, DODM starts to search for an appropriate organization that indeed meets the requirements that have been set. In this case, it first determines that the quality repair cost should go down:

```
internal(DODM)|active(CP1(ford, ford_design), pos)
```

This is further refined to the point of the introduction of lean production within the organization, which is one of the solutions to bring down the quality repair cost to 0:

```
internal(DODM)|active(CP2(ford, ford_design), pos)
internal(DODM)|active(CP3(ford, ford_design), pos)
```

As a result of this choice to introduce lean production, many changes in the current Ford design are sent to the output of DODM. First of all, the quality repair group is deleted:

```
output(DODM)|in_selected_DOD_output(ford_design,
                                    exists_group(quality_repair_group), neg)
```

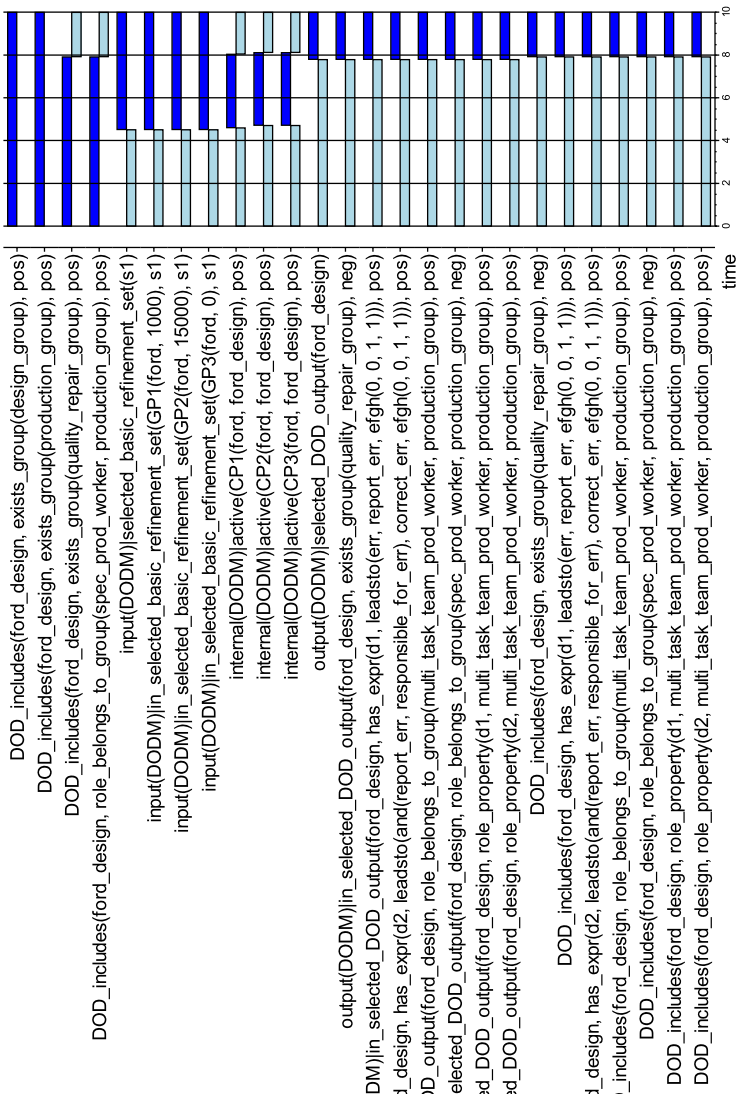


Fig. 11 DODM reasoning process

Furthermore, the specialized production worker role within the production group is deleted as well:

```
output(DODM)|in_selected_DOD_output(ford_design,
    role_belongs_to_group(spec_prod_worker, production_group), neg)
```

As a replacement for the specialized production workers, multi-task production workers are inserted into the organization.

```
output(DODM)|in_selected_DOD_output(ford_design,
    role_belongs_to_group(multi_task_prod_worker, production_group), pos)
```

Of course the behavior of this role is completely different from the behavior of the classical specialized production worker role. Since the approach which is used throughout the paper also allows for the specification of behavior of the roles, this behavior is also present on the output of DODM.

```
output(DODM)|in_selected_DOD_output(ford_design,
    role_property(d1, multi_task_prod_worker, production_group), pos)
output(DODM)|in_selected_DOD_output(ford_design,
    role_property(d2, multi_task_prod_worker, production_group), pos)
```

The actual behavior expected of an agent allocated to such a role is communicated in the form of a *leads to* property as introduced in the beginning of this section.

```
output(DODM)|in_selected_DOD_output(ford_design,
    has_expr(d1, leadsto(err, report_err, efgh(0,0,1,1)), pos)
```

This first property states that if an error is observed this error should be reported immediately. The second role property is specified as follows:

```
output(DODM)|in_selected_DOD_output(ford_design,
    has_expr(d2, leadsto(and(report_err, reposible_for_err),
    correct_err, efgh(0,0,1,1)), pos)
```

Stating that if an error is reported, and the worker is responsible for this error, he should correct the error immediately. Both properties are typical for the lean production system. Note that communicating such properties requires properties about properties, i.e. a meta-language in this case called meta-TTL. Finally, after all this has been sent to the output, the actual DOD is updated which eventually results in a restored demand again, as already shown in the high-level trace presented in Sect. 9.1.

10 Verification

To see whether the properties as expressed in Sect. 8 hold for the simulation trace, first of all, the RQSM_generate and DODM_generate properties have been checked against the trace shown in Fig. 7 using a software tool called the TTL Checker (cf. Jonker and Treur 2002; Sharpanskykh and Treur 2010; Bosse et al. 2009). Both properties were shown to hold for the trace.

In order to see whether the refinement process within RQSM is properly performed, the tree used for the simulation as presented before in Sect. 6 has been formally proven by means of the SMV model checker (McMillan 1993). The translation of the properties expressed in Sect. 6 to the input language of SMV is not trivial. In order to improve the efficiency of the checking process, the numbers as introduced in the case study above have been divided by 1000. In order to verify whether the property hierarchy is indeed correct, four rules have been specified in the SVM input language. The first rule concerns the calculation of the average price of cars on the market, which is simply calculated by adding the average design cost, production cost, and quality repair cost:

```
next(average_car_price) := average_design_cost + average_production_cost
                           + average_quality_repair_cost;
```

Furthermore, the calculation of the price of Ford is also specified in the same fashion as the calculation of the average price for cars with one intermediate step, namely the cost price. In this case the two are considered to be equivalent.

```
next(ford_cost_price) := ford_design_cost + ford_production_cost
                         + ford_quality_repair_cost;
next(ford_price) := ford_cost_price;
```

Final element is the calculation of the demand for Ford cars, which is directly coupled to the cost price. Notice that the calculation presented here are identical to the ones used in the simulations.

```
next(ford_demand) := (2000 * 4 * average_car_price) / ford_price;
```

Now finally, two checks are performed after having inputted the initial facts based on the scenario as used in the simulation and the transition rules as specified above. These checks are specified in CTL. The first one states that if the costs at the lowest level of the Ford organization are all equal all lower to the average costs over all companies, demand for Ford cars will be at least equal to a quarter of the total demand (constant at 2000):

```
AG (((ford_design_cost <= average_design_cost) &
   (ford_production_cost <= average_production_cost) &
   (ford_quality_repair_cost <= average_quality_repair_cost))
   -> AX(ford_demand >= 500))
```

A second version is a stronger requirement. It states that if the sum of the costs of all different groups is lower or equal to the average, demand will be at least a quarter of the total demand:

```
AG (((ford_design_cost + ford_production_cost + ford_quality_repair_cost) <=
   (average_design_cost + average_production_cost + average_quality_repair_cost))
   -> AX(ford_demand >= 50))
```

Indeed, both properties are satisfied given the initial conditions and the rules specified. Besides checking whether the lowest level properties satisfy the highest level

property, each of the interlevel relationships have also been checked in a similar manner, and were all shown to hold. Furthermore, to prove the successfulness of DODM, the property hierarchy shown in Fig. 6 has also been proven by the SMV model checker which shows that introducing lean production in a design object indeed results in canceling the quality repair costs, which satisfied the property DODM_successful. Two input rules have been specified, first of all, the definition of lean production, and secondly the effect of lean production (i.e. 0 quality repair cost):

```
next(production_method) := case
    !q_r_group & multi_task_team_prod_worker & multi_task_team_prod_worker_beh &
          !spec_prod_worker: lean;
    1 : mass;
          esac;

next(ford_design_cost):= case
    production_method = lean: 0;
    1: 4;
          esac;
```

The following property has been shown to hold:

```
AG ((!q_r_group & multi_task_team_prod_worker & multi_task_team_prod_worker_beh &
          !spec_prod_worker)
-> AF (ford_design_cost = 0))
```

In other words, for all time points, in case CP4-5-6 are indeed accomplished this reduces quality repair cost to 0, which clearly satisfies property CP1. Again, the intermediate relationships have been checked as well, and all were proven to hold. As a result, the DODM_successful property is satisfied as well as the RQSM_successful property in case the components indeed generate the output based on these property hierarchies.

11 Discussion

Organizations aim to meet their organizational goals. Monitoring whether events occur that endanger fulfillment of these goals enables organizations to consciously adapt and survive. Adaptation is essential once an organizational goal becomes unreachable. This paper views such a change as a (re)design process. A component-based formal generic model for design developed within the area of AI and Design is specialized into a model for organization (re)design. Such an approach can be used to support change managers in signaling when change is needed and in what direction to change the organization.

Formalizations developed within the area of Organization Theory and AI (or computational organization theory), have proved suitable for the description of organization models as design object descriptions, and organization goals as design requirements. Furthermore, different types of specialized knowledge have been identified: (1) about main organization goals and their relation for given environmental conditions to organization requirements, (2) about refinement of organization requirements, (3) about design object descriptions, and (4) which components for a design

object description satisfy which requirements. The generic design model was instantiated with such types of knowledge to constitute a specialized component-based model for (re)design of organizations. Example properties have been taken from a well known example in Organization Theory describing the introduction of lean production within an organization (Womack et al. 1991).

This paper focuses on external triggers for organizational change. Triggers are related to specific goals that play the role of design requirements which the organizational change should comply to. These requirements tend to be high-level goals and lack the detail needed for specifying how an organization should change. Therefore, design requirement refinement is introduced in the form of hierarchies of requirements. Such hierarchies relate objectives of the organization (e.g., high demand for cars) to organizational change properties at different organizational levels (e.g., change in some departments). Thus, they relate triggers at the level of the organization to properties at the level of parts (groups) within the organization. For example, the cause of why a certain type of car is not selling according to the goals that have been set is related to the costs of quality repair. Requirements hierarchies help to localize where to change the organization. High-level goals for an organization as well as goals for organizational redesign have been related to low-level executable properties. Formal verification has been performed and the results show satisfaction of the non-leaf properties in the property tree. Note that this shows that the approach itself has been formalized in an appropriate manner; an actual validation of the approach is beyond the scope of this paper. However, it does give a good idea of the possible success of the approach given that the organizational templates and the requirements that are used within the system are suitable. Burton and Obel (1995) raise the issue of validity of the organizational models designed from a computational perspective. They state that in order to create a valid model, three elements should be kept in balance: the question of purpose (for what purpose is the model being designed), the experimental design (how can the computational model be manipulated so that the purpose can be met), and the computational model itself (how has one chosen to define the model). Also in this paper an attempt has been made to keep all these elements in balance.

The main differences between the proposed approach and several approaches in the domain of Artificial Intelligence and computational models for organizational design have already been discussed in Sect. 2. When comparing the approach to previous work in the redesign of organizations the main strength is the formal description of the whole redesign process in terms of a generic redesign model for organizations. In the field of management for example, an overview of which can be found in Douglas (1999), only informal descriptions are given about redesign processes. In Systems Theory, see e.g. Rapoport (1986), goal oriented behavior is addressed. The gap observed between the actual state of the system and the desired state causes redesign, which corresponds with the approach taken in this paper. Formalizations by means of property hierarchies are, however, not present, therefore formal verification as done in this paper cannot be performed.

In Horling et al. (2001) a general diagnosis engine is presented which drives adaptation processes within multi-agent organizations using the TAEMS modeling language as the primary representation of organizational information. In the design of the

diagnostic engine three distinct layers are identified: symptoms, diagnosis, and reactions which in the approach presented in this paper roughly correspond to Sects. 5, 6, and 7 respectively. The implementation of these elements differs in both approaches. The goals and requirements in this paper are explicitly connected to each other. Once an organizational goal is observed not to be fulfilled, such a dissatisfaction is related directly to a goal for change. In the approach presented in Horling et al. (2001) lacks such an explicit relation between goals and error diagnosis. Furthermore, this paper also introduces an approach to diagnose whether the whole reorganization process was successful, which is not the case in Horling et al. (2001). Dignum et al. (2004) explores dynamic reorganization of agent societies and focuses on changes to the structure of an organization, this paper presents an approach that enables such a dynamic reorganization.

In Ishida et al. (1990) an approach is introduced which aims to archive adaptive real-time performance through reorganizations of the society. As a domain of application, production systems are used throughout that paper. Whereas that paper focuses on adaptive agents, this paper concentrates on adaptation of an organizational model that abstracts from agents and specifies elements on the level of roles the agents can fulfill.

The work presented in this paper can also be compared with the work on institutions as a way to describe multi-agent organizations. In Esteva et al. (2002) an institution is said to structure interactions and enforce individual and social behavior by obliging everybody to act according to norms. In that same paper, a formalization language is introduced for such an institution. The approach to use dynamic expression as a restriction of the behavior of agents allocated to that role used in this paper is also expressive enough to describe such norms. For example, in McCallum et al. (2005) an example of a norms is said to be the following: "Students are prohibited from sitting the exam if they have not completed the assignment" such can easily be formulated in terms of a dynamic property for the student role. The approach presented in this paper could therefore also be applied to institutions and normative organizations.

Finally, in the field of coalition formation (see e.g. Klusch and Gerber 2002; Shehory and Kraus 1995), the main purpose of forming a coalition is to perform a task that cannot be performed by a single agent. That work can be combined with our approach by addressing the problem of the allocation of agents to roles, after the change of the organizational model by our approach.

References

Ashworth MJ, Carley KM (2007) Can tools help unify organization theory? Perspectives on the state of computational modeling. Comput Math Organ Theory 13:89–111

Axelrod R (1997) The complexity of cooperation: agent-based models of competitions and collaboration. Princeton University Press, Princeton

Bashein ML, Marcus ML, Riley P (1994) Business process reengineering: preconditions for success and failure. Inf Syst Manag 9:24–31
Bosse T, Jonker CM, van der Meij L, Treur J (2007) A language and environment for analysis of dynamics by simulation. Int J Artif Intell Tools 16:435–464
Bosse T, Jonker CM, van der Meij L, Sharpanskykh A, Treur J (2009) Specification and verification of dynamics in agent models. Int J Coop Inf Syst 18:167–193
Bosse T, Jonker CM, Treur J (2010) Formal analysis of design process dynamics. Artif Intell Eng Des Anal Manuf 24:397–423. Preliminary version In: R Lopez de Mantaras, L Saitta (eds) Proceedings of the 16th European conference on artificial intelligence, ECAI'04, 2004, pp 293–297
Brazier FMT, van Langen PHG, Treur J (1998) Strategic knowledge in design: a compositional approach. Knowl-Based Syst 11:405–415
Brazier FMT, Jonker CM, Treur J (2002) Principles of component-based design of intelligent agents. Data Knowl Eng 41:1–28
Burton RM, Obel B (1995) The validity of computational models in organization science: from model realism to purpose of the model. Comput Math Organ Theory 1:57–71
Carley KM, Lee JS (2004) OrgAhead: a computational model of organizational learning and decision making. CASOS Technical Report CMU-ISRI-04-117
Carley KM, Svoboda DM (1996) Modeling organizational adaptation and a simulated annealing process. Sociol Methods Res 25:138–168
Carley KM, Kjaer-Hansen J, Newell A, Prietula M (1992) Plural-Sioar: a prolegomenon to artificial agents and organizational behavior. In: Masuch M, Warglien M (eds) Artificial intelligence in organization and management theory. Elsevier, Amsterdam, pp 88–118
Ciancarini P, Wooldridge M (eds) (2001) Agent-oriented software engineering. Lecture notes in computer science, vol 1957. Springer, Berlin
Cohen KJ, Cyert RM (1965) Simulation of organizational behavior, simulation of organizational behavior. In: March JG (ed) Handbook of organizations. Rand McNally, Chicago, pp 305–334
Dignum V, Sonenberg L, Dignum F (2004) Dynamic reorganization of agent societies. In: Proceedings of CEAS: workshop on coordination in emergent agent societies at ECAI 2004
Donaldson L (2001) The contingency theory of organizations. Sage, Thousand Oaks
Douglas C (1999) Organization redesign: the current state and projected trends, Manag Decis 37(8)
Esteva M, Padget J, Sierra C (2002) Formalizing a language for institutions and norms. In: Intelligent agents VIII. Lecture notes in artificial intelligence, vol 2333, pp 348–366
Ferber J, Gutknecht O (1998) A meta-model for the analysis and design of organisations in multi-agent systems. In: Proceedings of the third international conference on multi-agent systems (ICMAS'98). IEEE Computer Society Press, Los Alamitos, pp 128–135
Hall G, Rosenthal T, Wade J (1993) How to make reengineering really work. Harv Bus Rev 71(6):119–131
Hannoun M, Sichman JS, Boissier O, Sayettat C (1998) Dependence relations between roles in a multi-agent system: towards the detection of inconsistencies in organization. In: Sichman JS, Conte R, Gilbert N (eds) Multi-agent systems and agent-based simulation (proc of the 1st int workshop on multi-agent based simulation, MABS'98). Lecture notes in artificial intelligence, vol 1534. Springer, Berlin, pp 169–182
Hannoun M, Boissier O, Sichman JS, Sayettat C (2000) MOISE: An organizational model for multi-agent systems. In: Monard MC, Sichman JS (eds) Advances in artificial intelligence. Lecture notes in artificial intelligence, vol 1952. Springer, Berlin, pp 152–161
Haroud D, Boulanger S, Gelle E, Smith IFC (1994) Strategies for conflict management in preliminary engineering design. In: Proceeding of the AID 1994 workshop conflict management in design
Harrison JR, Lin Z, Carroll GR, Carley KM (2007) Simulation modeling in organizational and management research. Acad Manag Rev 32:1229–1245
Horling B, Benyo B, Lesser V (2001) Using self-diagnosis to adapt organizational structures. In: Muller JP, Ander E, Sen S, Frasson C (eds) Proceedings of the fifth international conference on autonomous agents. ACM Press, New York, pp 529–536
Hubner JF, Sichman JS, Boissier O (2002) A model for the structural, functional and deontic specification of organizations in multiagent systems. In: Proc. 16th Brazilian symposium on artificial intelligence (SBIA'02), Porto de Galinhas, Brasil. Extended abstract in: Castelfranchi C, Johnson WL (eds), Proc of the first international joint conference on autonomous agents and multi-agent systems, AAMAS'02. ACM Press, 2002, pp 501–502
Hubner JF, Sichman JS, Boissier O (2004) Using the Moise+ for a cooperative framework of MAS reorganization. In: Bazzan ALC, Labidi S (eds) SBIA 2004. Lecture notes in AI, vol 3171. Springer, Berlin, pp 506–515

Hyatt A, Contractor N, Jones P (1997) Computational organizational network modeling: strategies and an example. Comput Math Organ Theory 2:285–300
Ishida T, Yokoo M, Gasser L (1990) An organizational approach to adaptive production system. In: Proceedings of the 8th national conference on artificial intelligence, Boston, USA, pp 52–58
Jaffee D (2001) Organization theory: tension and change. McGraw-Hill, New York
Jin Y, Levitt RE (1996) The virtual design team: a computational model of project organizations. Comput Math Organ Theory 2:171–196
Jonker CM, Treur J (2002) Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. Int J Coop Inf Syst 11:51–92
Jonker CM, Treur J (2003) Relating structure and dynamics in an organisation model. In: Sichman JS, Bousquet F, Davidson P (eds) Multi-agent-based simulation II. Proc of the third int workshop on multi-agent based simulation, MABS'02. Lecture notes in AI, vol 2581. Springer, Berlin, pp 50–69
Klusch M, Gerber A (2002) Dynamic coalition formation among rational agents. IEEE Intell Syst 17(3):42–47
Kunz JC, Christiansen TR, Cohen GP, Jin Y, Levitt RE (1998) The virtual design team. Commun ACM 41:84–91
Lomi A, Larsen ER (2001) Dynamics of organizations: computational modeling and organization theories. AAAI Press, Menlo Park
McCallum M, Vasconcelos WW, Norman TJ (2005) Verification and analysis of organisational change. In: Boissier O, Dignum V, Matson E, Sichman J (eds) Proc. 1st OOOP workshop, pp 91–106
McMillan K (1993) Symbolic model checking: an approach to the state explosion problem. Kluwer Academic, Dordrecht
Rapoport A (1986) General system theory. Abacus Press, Cambridge
Sharpanskykh A, Treur J (2010) A temporal trace language for formal modelling and analysis of agent systems. In: Dastani M, Hindriks KV, Meyer JJC (eds) Specification and verification of multi-agent systems. Springer, Berlin, pp 317–352
Shehory O, Kraus S (1995) Task allocation via coalition formation among autonomous agents. In: Proceedings of IJCAI 1995, pp 655–661
Womack JP, Jones DT, Roos D (1991) The machine that changed the world: the story of lean production. HarperCollins, New York

Mark Hoogendoorn is an assistant professor at the Vrije Universiteit Amsterdam, Department of Artificial Intelligence. From August 2007 till September 2007 he has been a visiting researcher at the Department of Computer Science and Engineering of the University of Minnesota. He obtained his Ph.D. degree from the Vrije Universiteit Amsterdam in 2007. In his Ph.D. research he focused on organizational change within multi-agent systems, applying his research in projects in various domains, including incident management, logistics, and the naval domain. His current research interests include multi-agent systems, cognitive modeling, and ambient intelligence.

Catholijn M. Jonker is full professor of Man-Machine Interaction at the Faculty of Electrical Engineering, Mathematics and Computer Science of the Delft University of Technology. She studied computer science, and did her Ph.D. studies at Utrecht University. After a post-doc position in Bern, Switzerland, she became assistant (later associate) professor at the Department of Artificial Intelligence of the Vrije Universiteit Amsterdam. From September 2004 until September 2006 she was a full professor of Artificial Intelligence/Cognitive Science at the Nijmegen Institute of Cognition and Information of the Radboud University Nijmegen. She chaired De Jonge Akademie (Young Academy) of the KNAW (The Royal Netherlands Society of Arts and Sciences) in 2005 and 2006, and she is a member of the same organisation from 2005 through 2010. Her recent publications address cognitive processes and concepts such as trust, negotiation, and the dynamics of individual agents and organisations.

Jan Treur has a full professorship in Artificial Intelligence at Vrije Universiteit Amsterdam (VUA) since 1990. He is heading the department of Artificial Intelligence consisting of about 45 researchers. He is an internationally well-recognized expert in agent technology, cognitive modelling and knowledge engineering. He is or has been member of the programme committee of many of the main conferences and workshops and many journals in these areas. His extensive list of publications (e.g., see www.cs.vu.nl/~treur) covers major scientific publication media in artificial intelligence and cognitive science, including the top level conferences and journals. Some of his recent involvements are organising the International Workshop on Human Aspects in Ambient Intelligence.